ACKNOWLEDGMENT

Information on the Accura Portioning System was provided by Jon Hocker of FMC FoodTech, Inc. (fmcfoodtech.info@fmcti.com and http://www.fmcfoodtech.com). DSI Accura is a trademark of FMC FoodTech, Inc.

AUTHOR INFORMATION

Bonnie Heck (bonnie.heck@ece.gatech.edu) received the B.S. degree in electrical engineering from the University of Notre Dame in 1981, the M.S. degree in mechanical and aerospace engineering at Princeton University in 1984, and the Ph.D. degree in electrical engineering from Georgia Tech in 1988. She has been on the faculty of Georgia Tech since 1988, where she currently holds the rank of professor. She has also worked in industry for Honeywell Inc. as a design and test engineer from 1983 until 1985. She is active in scientific and professional societies, most specifically in the IEEE Control Systems Society.

How to Teach a Van to Drive

An undergraduate perspective on the 2005 DARPA Grand Challenge

JEREMY GILLULA and JEREMY LEIBS

It's the middle of summer in the Mojave Desert, and the heat is oppressive. The mercury easily tops 100 °F out among the scrub and the sagebrush, and there isn't a breeze to be found for miles. Add to these bone-dry conditions a sun so bright that even the shadows seem to have sought refuge elsewhere, and it's clear that anyone venturing into the depths of the desert on a day like this must be crazy. However, for the California Institute of Technology (Caltech) students slowly driving down an otherwise deserted road on the desert floor, it comes with the territory. They've been called a lot of things—including crazy—for trying to accomplish a task that many have told them is impossible. Their goal is to develop a robotic vehicle that can navigate completely autonomously through one of the most unstructured environments that exists: the Mojave Desert. The only help they are allowed to give their vehicle is a set of digital bread crumbs to follow in the form of waypoints that outline a corridor in which their vehicle is permitted to travel. Unfortunately, even that isn't much help since the corridor may be hundreds of feet wide and their vehicle must be smart enough to find a safe path, avoiding rocks, ditches, and the occasional cliff. Moreover, the vehicle must accomplish this task at an average speed of 20 mi/h. It's never been done before.

So far everything appears to be under control. Their vehicle, which they have dubbed "Alice," is in the lead position of a three-car caravan making its way down a desert road. Although Alice is not yet traveling at 20 mi/h and is swaying a little every now and then (a symptom of bugs yet to be worked out), she is making steady progress as she meanders onward. The students have attempted straight roads like this one before, but for the safety of the vehicle itself—not to mention the students riding along inside—they haven't let Alice enter more difficult territory. But that is about to change. They are now heading into the mountains.

Four hours later, they're deep inside the foothills of the Sierra Nevada, and Alice is still driving (see Figure 1). They've gotten past the windy roads with a little cheating (they let Alice control the steering while a human controlled the throttle for safety) and are now entering an area of slowly rolling hills. They've noticed a few bugs but decide that the terrain is safe enough to return full control to Alice. There's a certain thrill that comes with pushing a system that you've designed to its limits, especially when you trust that system with your life. Riding in Alice is almost like riding in a roller coaster, except there's no track telling you where you're going next; the vehicle makes that decision on its own. They crest another hill, drive to the bottom, crest another hill, drive to the bottom—and then it happens.

FIGURE 1 Alice. Originally a Ford E350, Alice was heavily modified by Caltech students to drive autonomously in off-road environments. The students chose to use a large van so that they could sit inside the vehicle as it was driving, enabling them to debug the system in real time.

FIGURE 2 Alice's desert battle damage. A bug in her software caused Alice to perceive a gently sloping hill further down the road as a sheer wall, and so she swerved into this berm to avoid it. The tire was shredded and the wheel was damaged. It would not be the last setback Alice or the students would suffer.

FIGURE 3 Alice's interior. In addition to being off-road capable, Alice had four workstations (two of which are pictured here), which allowed students to debug in real time. Some of the students ended up spending more time at these workstations than at any other location over the course of the summer.

Alice suddenly cranks the steering wheel hard left toward the edge of the road. The student in the driver's seat almost instantly hits the kill switch but not before the loud clang of metal on rock shakes the vehicle. When the dust settles, the rattled students look around, realize nobody is hurt, and let out a relieved laugh. It isn't the first time (nor would it be the last) that Alice has forcefully collided with something. When they get out to inspect the damage, they discover that what just happened was no simple fender bender, however. Alice has dug herself into a berm on the side of the road, and her front left tire is completely shredded. What's left of the tire is hanging loosely from the wheel, which is broken (Figure 2), and the steering drag link has been bent. Alice won't be driving again anytime soon.

Before the day is out, though, Alice will not have been the only vehicle to suffer the wrath of the Mojave. In an effort to obtain tools and bring a spare tire to Alice, four different support vehicles will be stuck simultaneously in a sand pit for nearly ten hours under the hot desert sun. Most of the students will suffer from mild dehydration; two of them will suffer from heat stroke. It will be four in the morning before the last group of students makes it home from the desert, tired and beaten. Nobody told them it was going to be like this when they signed up for the challenge. Then again, it probably wouldn't have made a difference—they would have volunteered anyway.

THE VOLUNTEERS

It has been said that the best results come from a volunteer work force. After all, volunteers are more dedicated to their job simply because it is their choice to work, and they have a personal investment in what they are working on. Recruiting students to work on Alice was no different. The project was pitched to the students at an organizational meeting in the fall term of the 2004–2005 academic year by Richard Murray, professor of control and dynamical systems at Caltech. The objective was to put together a team for the DARPA Grand Challenge (DGC), a competition that was originally announced by the Defense Advanced Research Projects Agency (DARPA) in March of 2003. The goal of the DGC was to generate the technology necessary to build and program an unmanned ground vehicle that could travel through 130 mi of difficult desert terrain completely autonomously in under ten hours. The first competition, held in March 2004, ended poorly. Although 15 teams qualified for the race (including an earlier incarnation of a Caltech team), none even came close to completing the course. The competition was to be held again in October 2005, and Caltech planned to field another entry. The project would be the challenge of a lifetime in the form of a multidisciplinary systems engineering class combined with a summer research program that was open to Caltech students in all fields of study. Caltech students, legendary for their desire to take on the impossible, started signing up (see "Team Caltech").

The students working on the project were divided into three main teams: the terrain team, the planning team, and the vehicle team. The terrain team was responsible for the vehicle's sensors and the associated software, including everything from pointing the cameras and laser range finders, to estimating the vehicle's position and orientation, to creating software that would map the terrain around the vehicle. This information was then passed to the planning team, which was responsible for planning a safe course through the vehicle's environment and then commanding the actuators to make the vehicle follow that path. The actuation was the domain of the vehicle team, whose responsibilities also included the day-to-day maintenance of the vehicle, hardware mounting, and power generation.

The students didn't have to start from scratch, though. Before Alice there was Bob, the vehicle Caltech entered in the first DGC. Bob (a 1997 Chevrolet Tahoe) had been cobbled together, to put it mildly. His interior was a rat's nest of wires and cables, and the software was not much better. Bob ended the first DGC stuck on a barbed wire fence after running off the road due to an error in his sensing systems. While that outcome provided the team with a menagerie of lessons about what to change the second time around, the most important lesson the team learned came long before the actual race: to win, they would have to be able to easily debug the system they had created. Of course, debugging such a complex system would be far easier to do on live data, since it would take them just as long to develop a simulation of the vehicle as it would take to build the vehicle itself. But at the same time, it was clear that it would be impractical to transmit over an existing inexpensive wireless network the huge quantities of data that some of the systems would be processing. To the students, the obvious solution was to seat as many programmers as possible inside the vehicle so that they could look at the data directly. Given that the vehicle also needed to be capable of off-road operation, there was only one logical choice: Alice.

ALICE

When the students originally saw Alice sitting in their shop, the first thing that struck them was how enormous she was. Originally a Ford E-350 van, she had been heavily modified for off-road use by Sportsmobile West, Inc. She featured a four-wheel-drive system with a geared transfer case, a reverse shackle design in the implementation of the front leaf spring suspension, and over 1.5 ft of suspension travel. For power, she had a diesel engine with a 46-gal fuel tank and a 3-kW generator, which directed power through two 1,500-W inverter/chargers into four 210-Ah marine gel-cell batteries. She could seat four, and each seat had a five-point racing harness to keep students safely strapped in. Coupled with a computer terminal at each seat and rack space for up to 16 servers (Figure 3), Alice was essentially a mobile computing lab packed into a van ready to tackle difficult desert terrain.

But when she was first delivered, that's all Alice was: an incredibly capable, but incredibly nonautonomous, van. Before she could start driving on her own, she needed the ability to control her own movement. For the first quarter of the school year, the vehicle team worked constantly on the task of actuating Alice. Five systems needed to be modified to realize computer control. Brake, throttle, and steering actuators were implemented first since they were critical for driving. Later, the ignition and transmission systems were actuated to deal with error modes, should the engine die or Alice need to reverse out of a dead end. Some of the systems were easy to implement. Alice's throttle was already electronic, so all the students had to do was tap into the existing system. Actuating the steering was just as simple; by attaching a gear to the steering shaft, the steering wheel could be controlled by using a servo motor. The brakes proved slightly more challenging. The team had encountered difficulty using a linear actuator on Bob, so for Alice they purchased a pneumatic system. This system ran

Team Caltech

Team Caltech's membership list frequently changed as new students came in and old ones graduated. This list is a snapshot of the team during the summer of 2005.

Graduate Students
- Lars Cremean, Caltech
- Kirsto Kriechbaum, Caltech
- Sam Pfister, Caltech
- Dima Kogan, Caltech

Undergraduate Students
- Lyudmil Antonov, VA Tech
- Henry Barnor, Caltech
- Ryan Cable, Caltech
- Eric Cady, Caltech
- Joe Donovan, Caltech
- Ken Fisher, Caltech
- Laura Fishman, Caltech
- Tully Foote, Caltech
- Jeremy Gillula, Caltech
- Marie Goransson, Lund Unversity
- Rob Grogan, Caltech
- George Hines, Caltech
- Erik Johannesson, Lund University
- Tony Kelman, Caltech
- Jeff Lamb, Caltech
- Jeremy Leibs, Caltech
- Gustav Lindstrom, Lund University
- Laura Lindzey, Caltech
- Lisa Nystrom, Lund University
- Harmon Pollock, Caltech
- Tami Reyda, Caltech
- Dave Rosen, Caltech
- Alex Stewart, University College, London
- Chris Wetzel, Caltech
- Lusann Yang, Princeton
- Jason Yosinski, Caltech

Special Thanks To
- Prof. Joel Burdick, Caltech
- Prof. Richard Murray, Caltech
- Prof. Pietro Perona, Caltech
- Prof. Christopher Rasmussen, University of Delaware
- Tony and Sandie Fender, Caltech

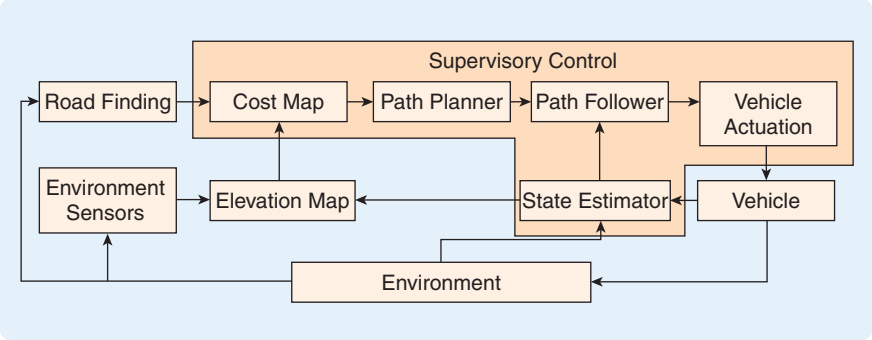

FIGURE 4 Alice's system architecture. The software that enabled Alice to maneuver autonomously features a complex set of interconnected software modules, each with its own inputs, outputs, and processing tasks.

on compressed air from Alice's onboard air compressor, allowing the vehicle to apply the brakes quickly and as hard as necessary. By winter break, the team had installed the systems necessary for basic operations, and, for the first time, Alice began to drive on her own.

While the vehicle team worked on actuation, the other two teams scrambled to endow Alice with a minimum level of functionality, and in the process she entered a somewhat grotesque Frankenstein phase. Everything other than the most critical actuators was slapped together at the last minute. Makeshift sensor mounts constructed from two-by-fours and a few bolts held a Navcom global positioning system (GPS) antenna and a laser range-finder unit to the roof, the Northrop Grumman LN-200 inertial measurement unit (IMU) was held to the floor of the van with little more than duct tape, and the software (assorted legacy code from Bob) was just as bad. Anything that could be run natively on the new bank of servers was simply copied over from the previous vehicle. Minimalistic code was written so that, to the old software, the new sensors looked like their historic counterparts. Different pieces of code sent and received messages using both old and new messaging architectures to maintain compatibility with all of the modules. It was as if Bob had entered the lair of a mad scientist and emerged as a schizophrenic van named Alice.

As would be expected, Alice's first tests were not very inspiring. At the time, she was little more than an enormous remote-controlled car and an unreliable one at that. Her actuators could execute simple commands generated by human operators, but that was the extent of her autonomy. However, with much persistence throughout the winter term, Alice's driving gradually improved. To do this, the team had to overcome four critical challenges: localization, trajectory following, mapping, and planning, which when connected together would form Alice's complex software architecture (see Figure 4).

FOUR EASY STEPS TO AUTONOMOUS NAVIGATION

To determine her location as precisely as possible, Alice needs to combine the outputs of the GPS unit and the IMU. The outputs of the IMU's accelerometers and gyros are integrated numerically according to the equations of motion to produce an inertial navigation solution. However, since a numerically integrated solution is prone to accumulation of integration errors, this inertial solution must be corrected by the GPS measurements using a Kalman filter. Additional smoothing is needed to prevent jump discontinuities in the state estimate when corrections are applied. These discontinuities can lead to spurious obstacles in the map and sudden swerves by the trajectory follower. The result of the inertial navigation and correction is an estimate of Alice's northing, easting, altitude, roll, pitch, and yaw, as well as their first and second derivatives, along with a precise time stamp. These estimates are broadcast to the other modules at roughly 40 Hz so that every module in the system knows where Alice is at a particular instant in time.

Alice then needs to take the estimate of her location, compare it to the path she is supposed to be following, and command the actuators accordingly. At first her movements were jerky, her steering lurching back and forth across the road while simultaneously slamming on the brakes and gas. (One student compared Alice's driving to that of a drunken teenager, blind in one eye, learning to drive.) However, with a team of three students devoting all of their time to this trajectory follower, and a diligent safety driver ready to take control if necessary, Alice was soon following predefined paths using minimal steering effort and smooth acceleration. The implemented trajectory follower consists of two separate controllers: one for the lateral (steering) control and one for longitudinal (speed) control. Both are essentially programmable-integral-differential (PID) controllers with feedforward terms and integral resets to avoid oscillations. With additional refinement, the combined trajectory follower proved to be one of the most robust of the vehicle's components, able to track paths to nearly 20-cm accuracy. However, following a path is still a long way from being able to detect obstacles and avoid them while driving through the Mojave Desert. That is where the mapping and planning software comes in.

To sense her environment, Alice is equipped with three major types of sensors. The first type is a simple firewire camera, the images of which are analyzed using an algorithm developed at the University of Delaware to determine the presence and location of roads. The second type is made up of pairs of cameras coupled in what is known as a stereovision system, which reconstructs scenes in the same way as a pair of human eyes, using disparity between

images to perceive depth. The third type—laser range finders (also known as LADARs, for laser detection and ranging)—sweep laser beams out in front of the vehicle, measure the time for reflection and, using some simple geometry and trigonometry, report the shape and location of the surrounding terrain. In the beginning, Alice was equipped with only one LADAR unit, the output of which was transformed into an elevation map as Alice drove. In essence, this elevation map consisted of a grid of cells, and each cell was assigned a height corresponding to the height of the terrain that the vehicle had measured at that location.

However, many more sensors would be needed for race day, both as backups in the case of failures and to provide Alice with the ability to see more accurately at greater distances. The difficulty, however, lay in combining the maps each sensor created, especially in cases where two sensors reported different data about the same patch of ground. To circumvent this problem, a heuristic weighted averaging algorithm is used. This algorithm combines low-weight measurements from sensors that are pointed further away from the vehicle, and thus more prone to error, with high-weight measurements from sensors pointed closer to the vehicle and thus more likely to be accurate. Simply having a map of the terrain, however, is not enough. Alice needs a way to reason about her surroundings, to identify which terrain constitutes a hazard and which terrain is safe.

To identify hazardous terrain, another set of heuristics is used to analyze the maps, this time to identify characteristics such as roughness using a discrete lowpass filter and obstacle height, using the standard deviation of elevation estimates for a given cell in the map. The result is a single number for each cell in the map, indicating how fast Alice can safely drive over that cell (see Figure 5). A speed of zero indicates an obstacle, a cell over which Alice should not drive. For cells where no terrain data has been collected, a low speed is chosen on the off chance that the sensors simply missed a spot. As a result, the problem of how to choose the vehicle's intended path—the planning problem—can be formulated as a receding horizon control optimization problem. All Alice has to do is optimize the time it takes her to get from her current position to a point further down the course, while satisfying physical constraints that, for example, prevent her from taking corners too fast or from running into obstacles.

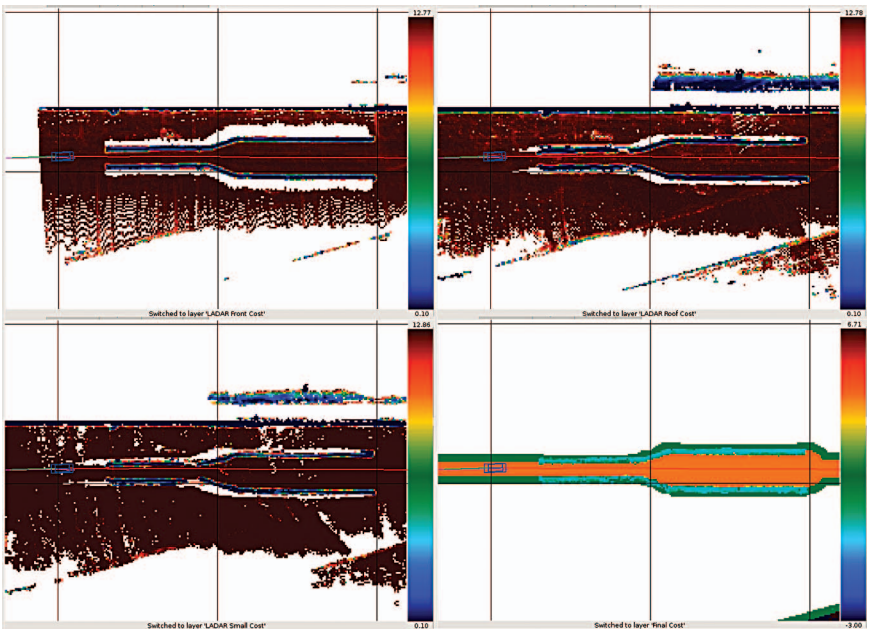

FIGURE 5 Sample elevation and cost maps. These diagrams indicate how Alice perceives the terrain around her. Each diagram shows data from a different sensor, where cells are color-coded according to the cost associated with each cell. The small blue rectangle to the left of each map is Alice, the red line is the path she has driven, and the green line indicates her direction of travel.

Like most theoretical solutions to practical problems, though, each of these four ideas (state estimation, trajectory following, mapping, and path planning) was easier to design than implement. The students had a long, hard road ahead of them before they would be confident in Alice's navigational abilities.

ALICE IN ACTION

After Alice was delivered and actuated, the time began to pass more quickly. Before they knew it, the team was already well into the spring term and the next big challenge was rapidly approaching: the site visit. DARPA had decided to perform testing of every vehicle that had entered the competition to reduce the number of teams that would make it to the next round, the National Qualifying Event (NQE), in September. If a team didn't pass the site visit, they were immediately disqualified. Fortunately, DARPA had announced the procedure for the inspections ahead of time. Each vehicle would have three chances to compete a simple zigzag course on a flat parking lot, in which the DARPA officials placed two garbage cans at random, so as to verify the vehicle's obstacle-avoidance capabilities. While the team knew the path-following software worked, they were worried about the obstacle avoidance. The week before the site visit degenerated into a frenzied delirium of running Alice, crushing garbage cans, fixing bugs, and crushing more garbage cans. The work continued around the clock, with some of the students

putting in as many as 100 h that week to make sure Alice was ready. Finally, the night before the site visit Alice was placed at the starting line. She drove down the course, approached the first trash can, slowed down, veered left, and continued onwards, taking the inside corners of the zigzag to maximize her speed until approaching the second trash can, steering around it and proceeding on to the finish line without incident. The students rejoiced, but in the back of their minds they wondered if she would be reliable enough to do the same thing three more times the following day.

The next day, the officials arrived. After a brief tour of Alice and her systems, the students once again brought her to the starting line. To reassure the officials that the vehicle was indeed driving autonomously, the students decided to run Alice without anyone in the driver's seat. It was the first time they had done so, and with good reason. If Alice made a wrong move, they would have to rely on the remote-controlled emergency-stop system to stop her before anyone was injured. Once the officials had placed the trash cans and all the systems were checked one last time, they began. Alice attempted the course three times but only successfully completed two of the runs. On the third run, the students had decided to increase her maximum speed in an effort to impress the officials, but the plan backfired. A bug in the mapping software registered flat ground midway down the course as an obstacle Alice could not drive around, and she was stuck. Although Alice still passed the inspection, the problem demonstrated to the team that Alice would need to be equipped with the ability to automatically recover from arbitrary failures, and as a result the supervisory control module (SuperCon) was born.

SuperCon was designed to monitor all of Alice's systems for fault conditions and then coordinate recovery. If Alice got stuck in front of an obstacle, SuperCon would observe that the vehicle was not making forward progress and would tell the path follower to back up so that the sensors could have another look at whatever was blocking her path. If the engine died, SuperCon would tell the actuation software to bring the vehicle to a halt and then restart the ignition. If the vehicle's position estimate proved to be completely wrong, SuperCon would tell the mapping software to erase the parts of the map that were generated using faulty data. The team planned for dozens of other different situations, many of them vague enough to catch any error short of the vehicle catching on fire. The students were determined not to let what happened to Alice during the third run of the site visit (and to Bob during the last race) happen again.

Despite the incomplete third run, DARPA selected the team to proceed to the qualifying event in September. Of course, to win the race, the vehicle needed to be able to drive in terrain far more challenging than a parking lot. The team continued working, and by June they believed they had a solid foundation. They were finally ready to take their creation out into the desert to discover its limits on challenging terrain. But it still came as no surprise to some of them when Alice crashed wheel first into the rocky berm. Combined with the additional support vehicle failures, the message the desert was giving them was clear: there was still a great deal of work to do.

SUMMER OF SETBACKS

The next ten weeks went by in a blur. A ridiculous series of failures hindered the team's progress at every step of the way. Between a failed air-conditioning compressor, a broken serpentine belt, a cracked heater core, a leaking power steering pump, and a defective alternator, the team spent a large fraction of the summer stranded in the shop, relying on simulations and logged data to further develop their software. In spite of these hindrances, the team persevered and continued working, taking Alice out to the desert when possible between different system failures. Gradually, Alice's performance improved, but as soon as one series of hardware failures was resolved, another would pop up. For example, one day when out testing, the vehicle spontaneously stalled and the engine would not start again. After 36 h and a US$500 tow back to Pasadena, the team finally found the culprit: a blown fuse that cost US$0.50 to replace. Although replacing the fuse got the vehicle to start, it did not fix the underlying problem. For the next several weeks the power continued to glitch, occasionally blowing the fuse and bringing Alice to a halt. Sometimes Alice would go for days with no problems, and then it would happen again. Ford technicians suspected a short in the wiring harness—a problem that would be nearly impossible to sort out given all of Alice's modifications. Some of the team was so discouraged they even discussed bringing Bob back out of retirement. Fortunately, when following Alice up a hill during a period of late-night testing, the driver of the chase-vehicle noticed a shower of sparks spray out the side of Alice as she went over a bump. They had found the short. One of the winch cables had come loose when replacing the bumper and was hanging down onto the tail pipe. It had taken a few weeks for the tail pipe to melt through the insulation, and thereafter the power was shorting when going over large bumps or up steep hills. Since the winch cabling wasn't needed, it was simply removed, and the vehicle was as good as new.

It was about this time—two weeks before the qualifying event—that things finally started coming together, almost miraculously. The vehicle stopped breaking down, software stopped crashing, and the speeds at which Alice was driving started to pick up. The team pushed even harder, knowing that to win, the vehicle would have to be foolproof as well as fast. They split into shifts so that Alice could be tested 24 hours a day. Some students began putting in 80-h work weeks, camping out in the Mojave so that they could be on hand to debug whatever problems might crop up. Alice's test runs continued to increase in length until, to the

student's surprise, the vehicle actually became somewhat reliable. The students started throwing more difficult challenges at her. With a little work, Alice could run for short periods without GPS. She could navigate precisely and repeatedly between two small obstacles on the road. She could take on rocky terrain but was smart enough to avoid it whenever possible. With four days to go, it seemed as though Alice could do just about anything. However, there was still one challenge left that the students knew they had to attempt, one of the most dangerous challenges they could think of. They needed to take Alice through Dagget Ridge.

Dagget Ridge, for those unfamiliar with the first DGC, represents some of the toughest terrain in the Mojave. Located near Barstow, California, the desert road running through Dagget Ridge is a twisty, rocky, unforgiving mountain pass that was the primary reason most vehicles didn't complete the first DGC. The route features the worst kind of desert terrain: sheer cliff faces, switchbacks, loose rocky roads, and blind curves. It was many times more challenging than the mountain roads the team had attempted at the beginning of the summer, but this time they had an entire summer's worth of experience and bug fixes under their belts. They figured they could pull it off. They had to, if they were to have any hope of winning the DGC. So they entered the waypoints from the previous DGC and set their course.

Put simply, it was terrifying. The slightest wrong move, a glitch in the sensors or a jump in the vehicle's estimate of its position, and Alice would have gone careening off the side of a cliff into the gully below, programmers and all. One of the programmers couldn't stand to watch. He knew the danger was very real. He had been in Alice during the first crash at the beginning of the summer. He knew what a mistake would mean. But what terrified him the most was that he was trusting his life to software he had helped write. And although he trusted it enough to stay in the vehicle, he still didn't hesitate to close his eyes and pray. When Alice suddenly turned to drive off a cliff, only the lighting fast reflexes of the operator in the driver's seat kept them from tumbling to their deaths. A bug in the mapping software had caused the problem. The speed assigned to areas of the map that had no associated sensor data (such as the open space off the edge of a cliff) was too high compared to some of the rocky terrain that made up the rough road Alice was supposed to follow. Thus, by Alice's logic, it would be faster to drive off the cliff than stay on the road. They adjusted the constants to make Alice stay on even the roughest of roads and then continued. By the time the programmer opened his eyes again, they were through. They had made it through Dagget Ridge, alive and intact.

HAY BALES, K-RAILS, AND THE RACE

With Dagget Ridge out of the way, the course DARPA had set up at the NQE should have been a breeze, but it was not so. Held at the California Speedway in Fontana, the NQE was essentially a snapshot of the racecourse itself, shrunk down to fit in the infield of a NASCAR racetrack and littered with the sorts of obstacles the vehicles could expect to encounter on the real course. There were abandoned cars to drive around, simulated cattle crossings to drive through, sheer cliffs to maneuver next to, bumpy terrain to overcome, an overpass to drive under, and of course, a tunnel designed purposely to block out GPS signals. On the first run, as Alice approached a narrow road lined with hay bales, a GPS registration error placed the course directly over the left row of hay. SuperCon jumped into action, brought the vehicle to a stop, and backed Alice up to see if there was another path around. There wasn't, and so Alice did exactly as she was designed to. She plowed straight ahead, easily driving her left wheel over the hay bales and continuing unfazed (Figure 6). However, it was the tunnel that presented the real problem. When she emerged on the far side, Alice had been without GPS corrections for too long. The state estimate failed to reconverge correctly, and Alice simply became progressively more confused until she had completely turned around on the course. The students found a fix, though, in the form of a zero-velocity update to further correct IMU integration errors. Some last-minute code was added to allow Alice to use the knowledge that she was stationary to zero out errors and refine her position estimate even without a GPS signal. With the last bug fixed, the next three runs proved successful, and Alice was selected as one of 23 teams to qualify for a position on the starting line. On 6 October, the team packed up their equipment and made the journey to Primm, Nevada, the site of the race. On 7 October, they went over Alice's systems one last time, ran through procedures, and mostly waited nervously. Then came 8 October.

FIGURE 6 Alice's first national qualifying event run. Due to a global positioning system registration error, Alice perceived driving over the hay bales as the only available option. Despite her somewhat destructive performance, Alice went on to complete the entire NQE course several times without incident, and was selected as a finalist to proceed to the starting line of the race.

FIGURE 7 Alice's final moments during the race. Due to an error in her state estimate, Alice crashed into a concrete barrier shortly before being disabled. Fortunately, Alice was mostly unharmed, and she continues to serve as a platform for a variety of research projects. (Photo © Will Heltsley.)

Finally, it was the day of the DGC—the day for which the team had prepared for nearly a year. A few on the team tried to sleep the night before, but no one was very successful. At 4 a.m. everyone was back in the pit area going over the route DARPA had distributed. They uploaded it to Alice's computers, confident enough in Alice's abilities that they made no changes to the file. Alice would be able to figure everything out on her own. They drove her to the starting chute, and with one last check of her systems, they closed the doors and stepped away. A few minutes later, the DARPA control team gave the okay for her to depart, and she was off. In the stands overlooking the starting area, a group of Techers—friends of those who had worked on the project—ripped off their shirts, stood up and cheered, letters painted on their chests spelling out CALTECH. The press photographers went wild, and Alice drove off into the distance.

The course had been structured to loop around on itself several times over, providing the media with many opportunities to snap photos and take videos of the vehicles as they sped past. One of those loops was the first eight miles of the course, running past the stands, out into the desert, and then down a straightaway pointed due south, next to a press viewing area. About 30 minutes after she had departed the starting area, Alice was spotted coming into view by that press viewing area, driving underneath a set of high-voltage power lines. But something was wrong. She stopped for a moment. In the postrace analysis, the students would discover that the power lines had interfered with the GPS signal, and Alice had brought herself to a stop to try to refine her position estimate. However, when she started again, the state estimate had not converged properly and there were still substantial errors. Alice decided to temporarily ignore the GPS signal, and although she thought she was driving due south, straight down the road, she was actually driving south-southwest, directly towards a set of concrete barriers set in place to protect the media from any errant vehicles. And then it happened. Again.

Alice crashed. This time, when the dust settled, nobody was laughing. Unable to correct for her state estimate problem, and hampered further by an unexplained failure of her medium-range LADAR units, Alice toppled and drove over one of the concrete K-rail barriers before her onboard error-detection systems brought her to a stop (Figure 7). Although there was no permanent damage (a testament to the strong front bumper provided by the Aluminess company), Alice was out of the race. It was over for Caltech, but five teams would go on to finish the course. Stanford, sponsored by Volkswagen, was the winner.

THE END

After the race, the team broke up quickly. Some of the students, who had come only for the summer, returned home to their native institutions. Others redirected their efforts toward graduating, trying to make up for the classes they had missed while focusing their time on the project. A few, not satisfied with the results of the DGC, have continued to work on Alice. In this respect, she has become an amazing research platform for use in several fields, including control theory. Currently, she is serving as the centerpiece of four separate thesis projects, as well as the basis of a project for an introductory control theory class. And although Alice may not have won the DGC, she did succeed in her original purpose: to teach a new generation of Caltech students about engineering, how to apply theory to the real world, how to debug and deal with shortcomings and schedules, and most importantly, how to work as a team on a complex problem. From the beginning, Alice was meant to be a vehicle for learning, and in that respect, she accomplished more for education than any van in Caltech history.

AUTHOR INFORMATION

Jeremy Gillula (jeremy@caltech.edu) is a senior majoring in computer science and minoring in control and dynamical systems at the California Institute of Technology. He has spent the last three years of his life working on Alice and her earlier incarnations. He plans to attend graduate school in the fall, where he will pursue research in sensing and perception for autonomous systems.

Jeremy Leibs is a student of computation and neural systems at the California Institute of Technology. Although originally a biologist, his interest in robotics and artificial learning led him to change his focus at the end of his sophomore year. Joining the DARPA team at the beginning of the 2004–2005 school year, he ended up responsible for Alice's state-estimation software and subsequently added a control and dynamical systems minor.